

Encontro 04 - Grafos e simulação de robôs móveis



INTELI

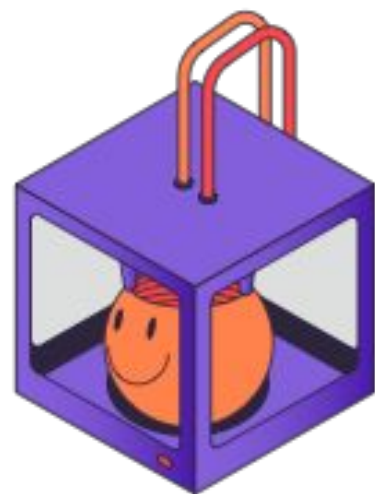
Engenharia de Computação
Módulo 6

Maio - 2023

Roteiro



- Atividade ponderada 2
- Árvores
- Grafos
- Representação de grafos
- Mão na massa - simulação de robôs



Encontros M6

Semana 1

 Introdução à robótica móvel

Semana 2

 Tipos de dados abstratos

Semana 3

 Estruturas de dados

Semana 4


 Simulação de robôs móveis

Semana 5


 Conceitos de visão computacional clássica

 Integração de robôs móveis

Semana 6


 Detecção de objetos com visão computacional clássica

Semana 7

 Detecção de objetos com visão computacional clássica

 Integração de sistemas 1

Semana 8

 Integração de sistemas 2

Semana 9

 Estudo de caso de implementação de simulação robótica

Semana 10

 Correção de problemas e refinamento

 Finalização do protótipo da solução

Árvores e Grafos



Árvores

- ❑ As listas ligadas usualmente fornecem maior flexibilidade que os arrays, mas são estruturas lineares, sendo difíceis de serem utilizadas para uma representação hierárquica.
- ❑ Pilhas e Filas refletem uma certa hierarquia, mas são limitadas a somente uma dimensão.
- ❑ Para superar esta limitação, utilizaremos um novo tipo de dados chamado **Árvore**. Uma árvore é formada por um conjunto de elementos denominados **nós** conectados através de **ramos** ou **arcos**.

Árvores

(a) é uma árvore vazia

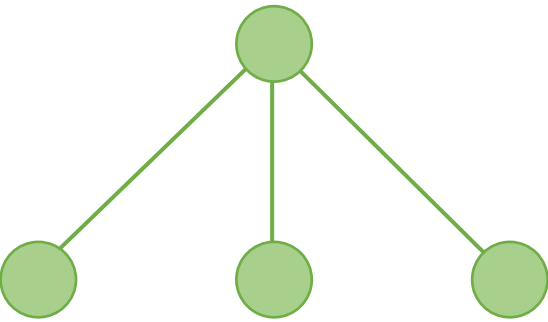
a)



b)



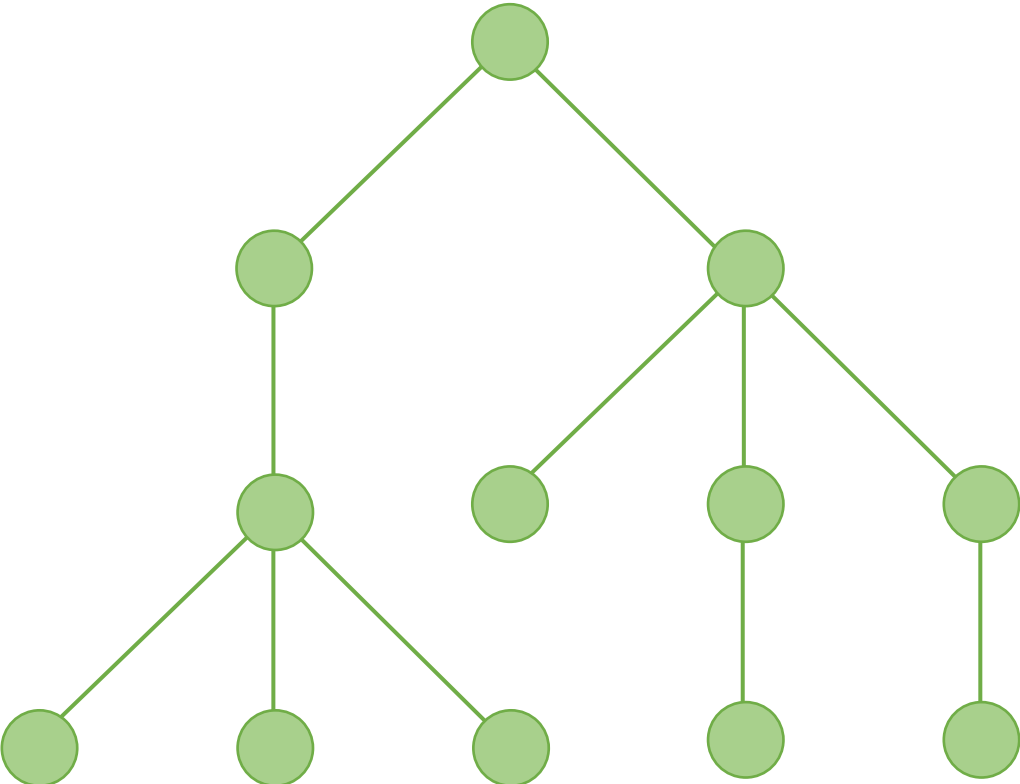
c)



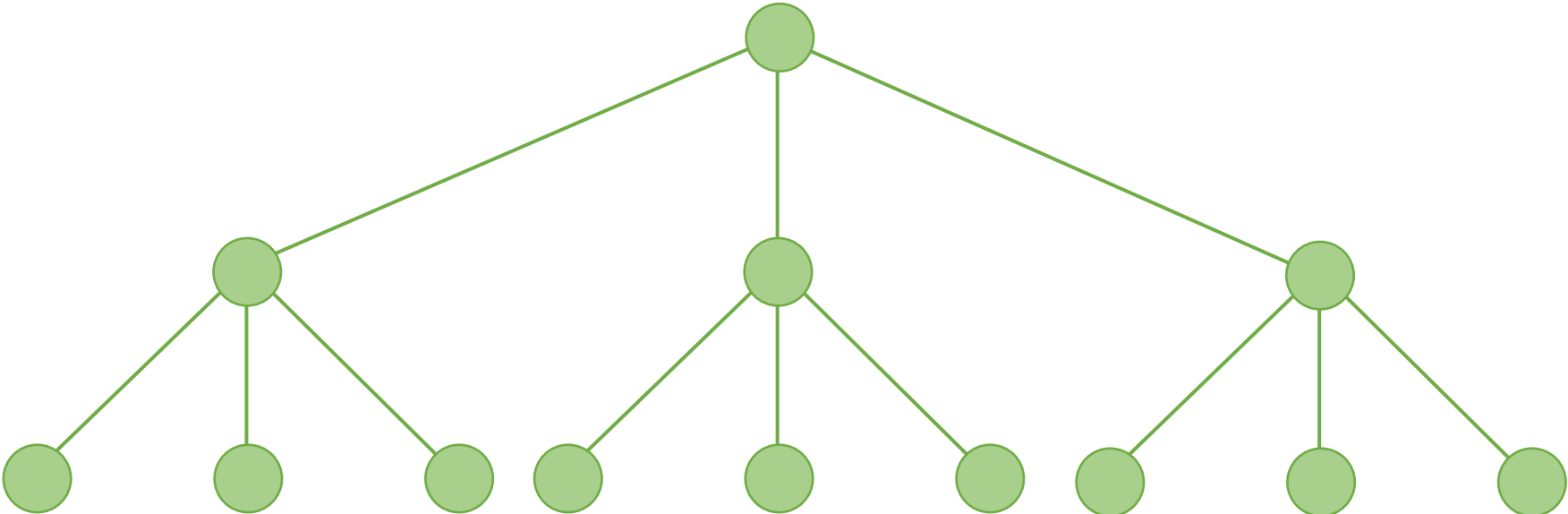
d)



e)



f)



g)

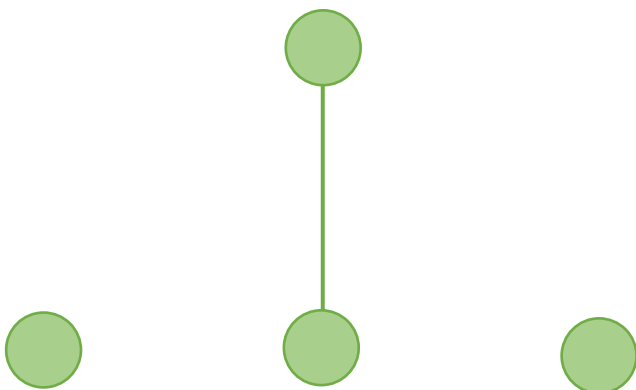
Grafos

- ❑ Em uma árvore, cada nó tem que ser atingível a partir da raiz através de uma sequência única de arcos, chamada de caminho. Desta forma, representam somente relações de um tipo hierárquico, como entre o ascendente e o filho.
- ❑ A generalização da árvore é o ***Grafo***, onde esta limitação desaparece. Com isso, temos que um grafo é um conjunto de vértices (ou nós) conectados por arestas (ou ramos).

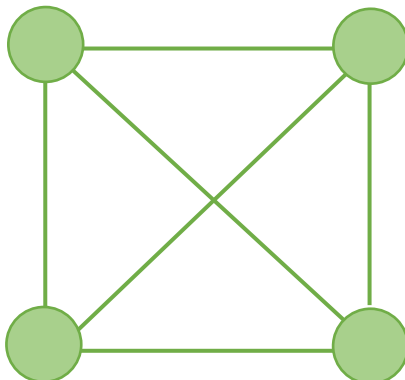
Grafos



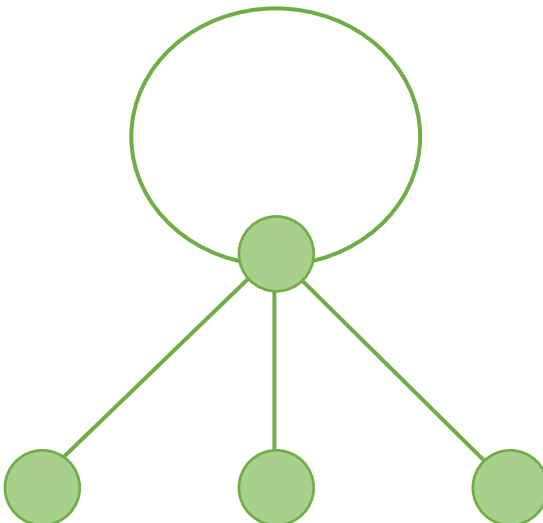
a)



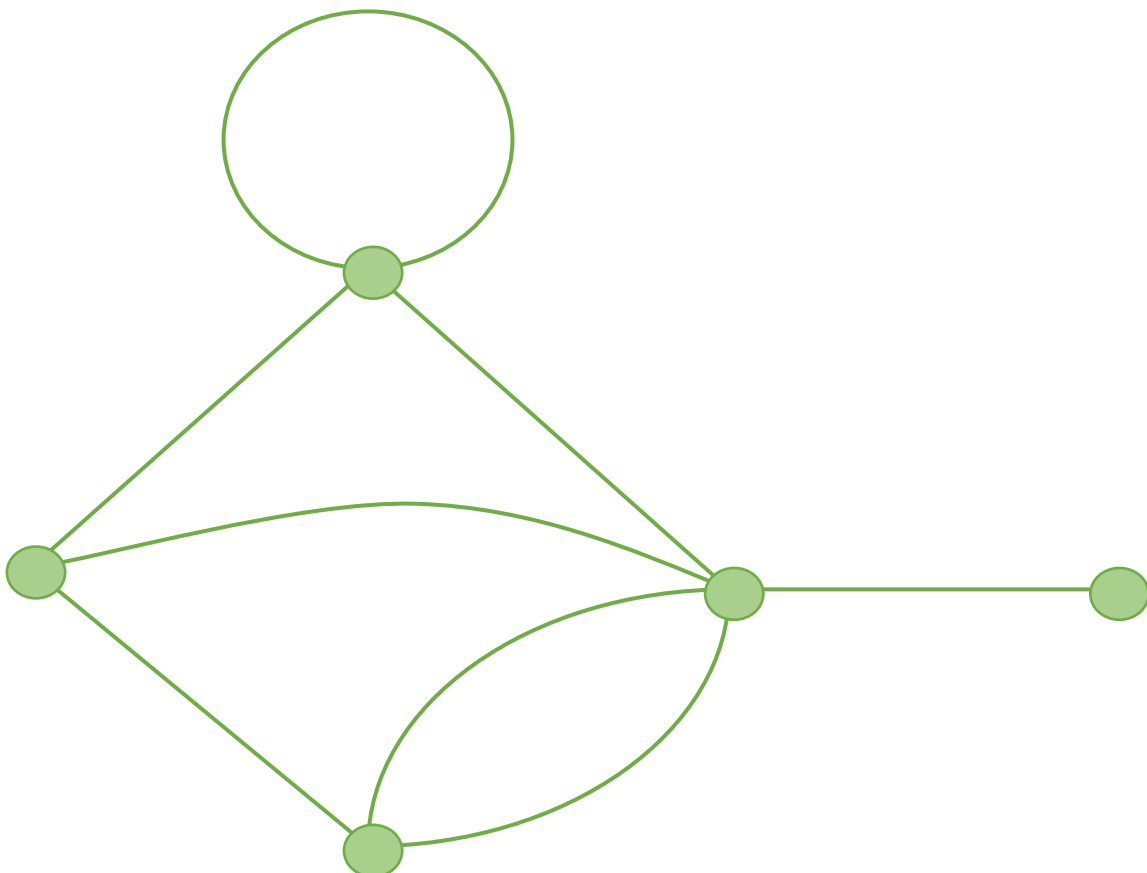
b)



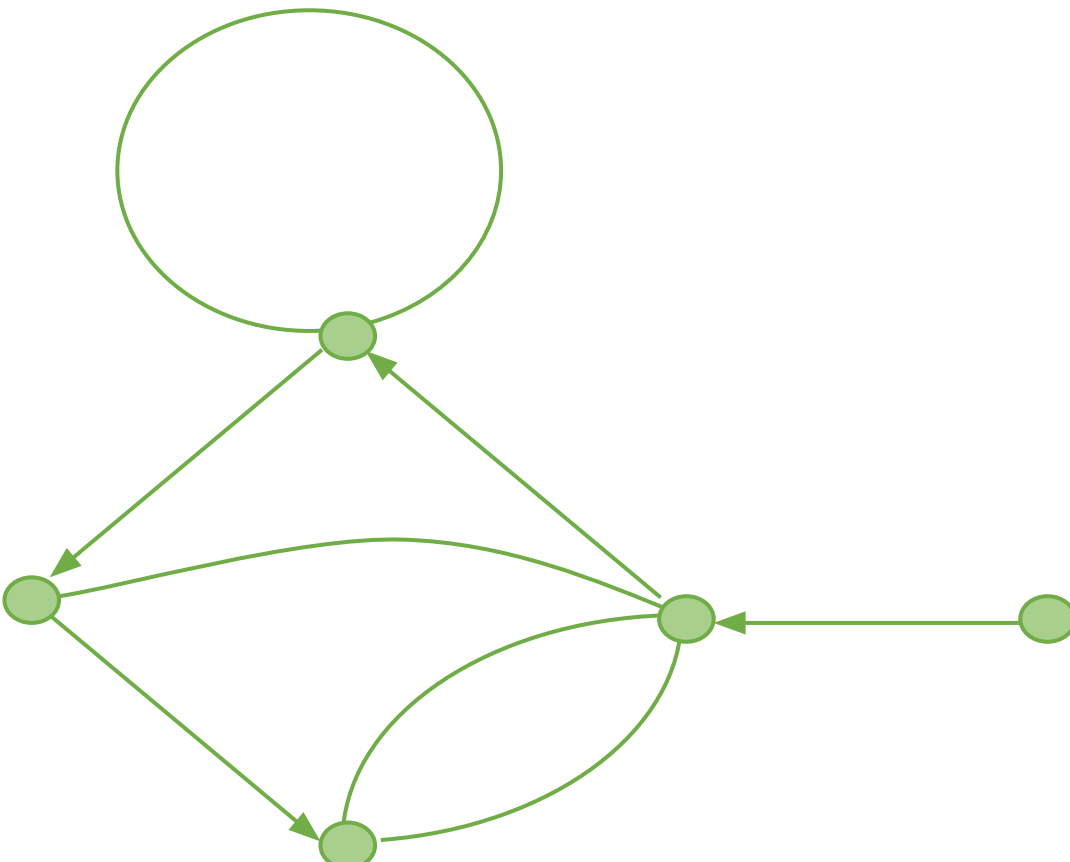
c)



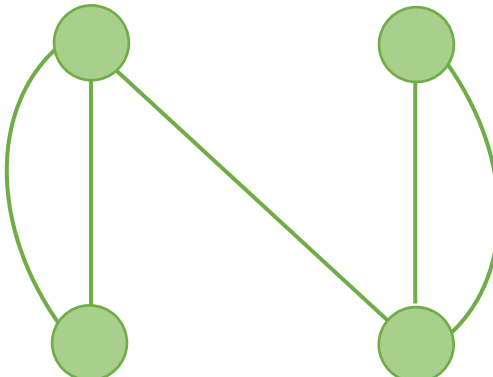
d)



e)

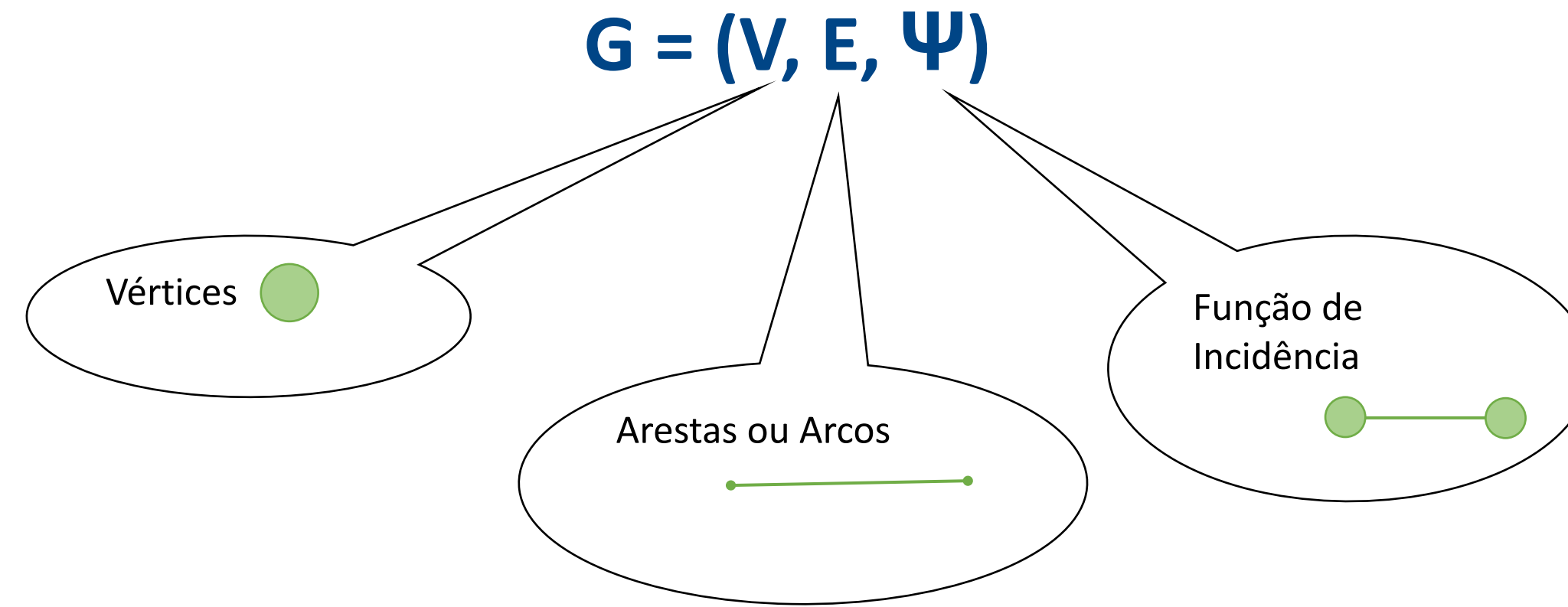


f)



g)

Grafos - terminologia



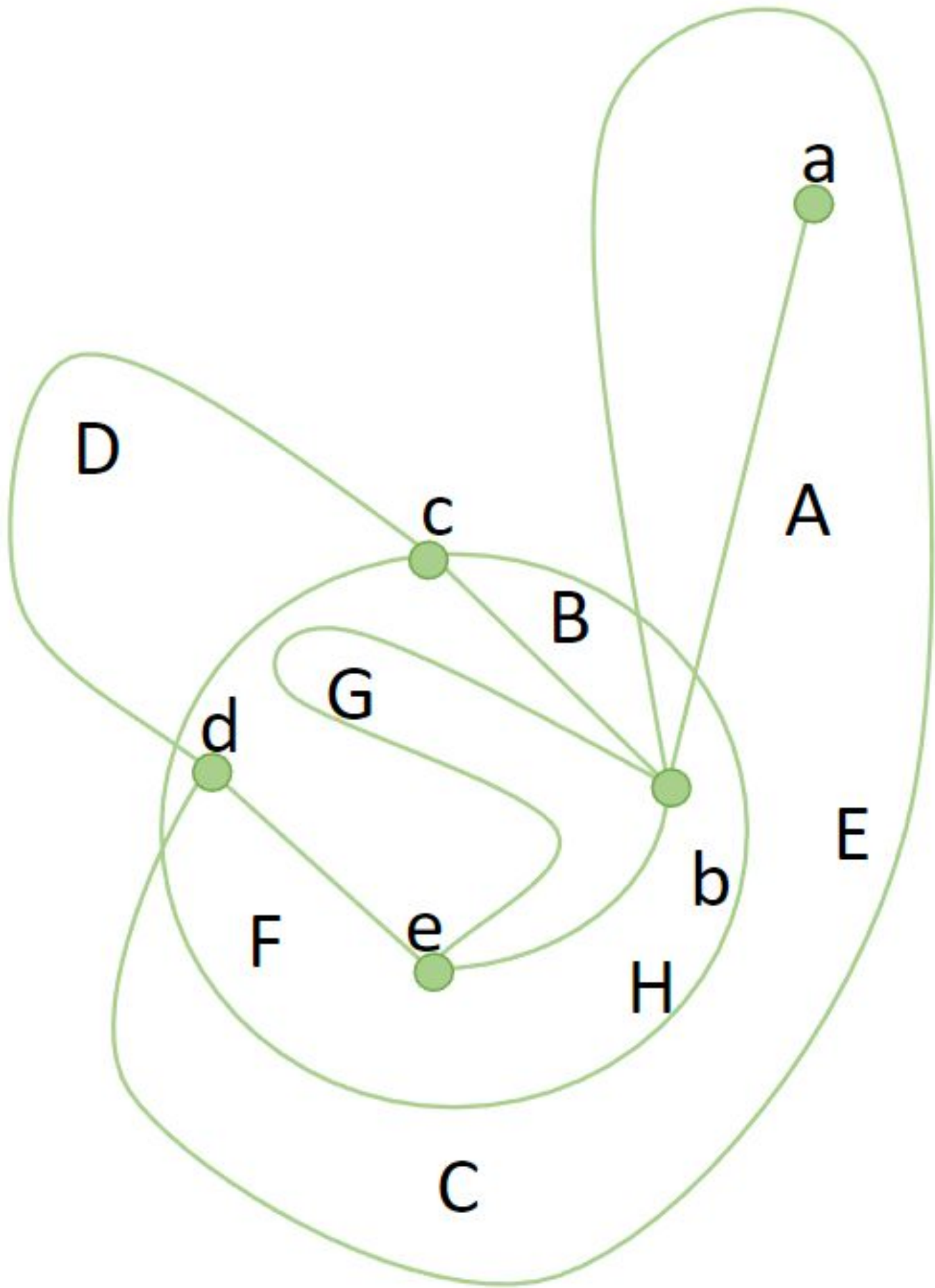
- ❑ Um grafo simples G consiste em um conjunto V não vazio de vértices e um conjunto E de arestas que pode ou não ser vazio. Cada aresta um conjunto de dois vértices.

Grafos - exemplo

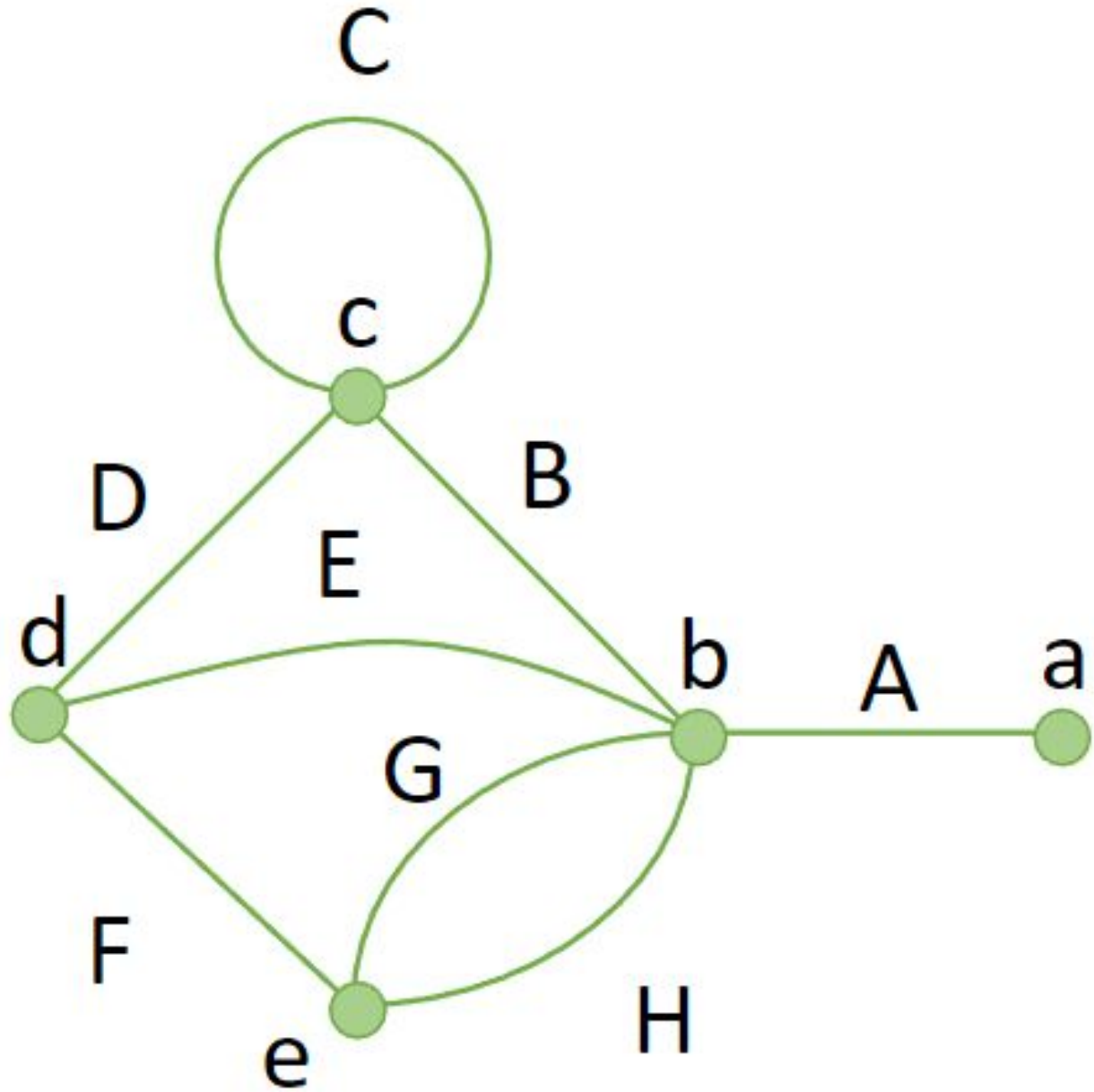
- $G = (V, E, \Psi)$
- $V = \{a, b, c, d, e\}$
- $E = \{A, B, C, D, E, F, G, H\}$
- Ψ :

$\Psi(A) = \{a, b\}$		$\Psi(E) = \{b, d\}$	
$\Psi(B) = \{b, c\}$		$\Psi(F) = \{d, e\}$	
$\Psi(C) = \{c, c\}$		$\Psi(G) = \{b, e\}$	
$\Psi(D) = \{c, d\}$		$\Psi(H) = \{b, e\}$	

Grafos - exemplo

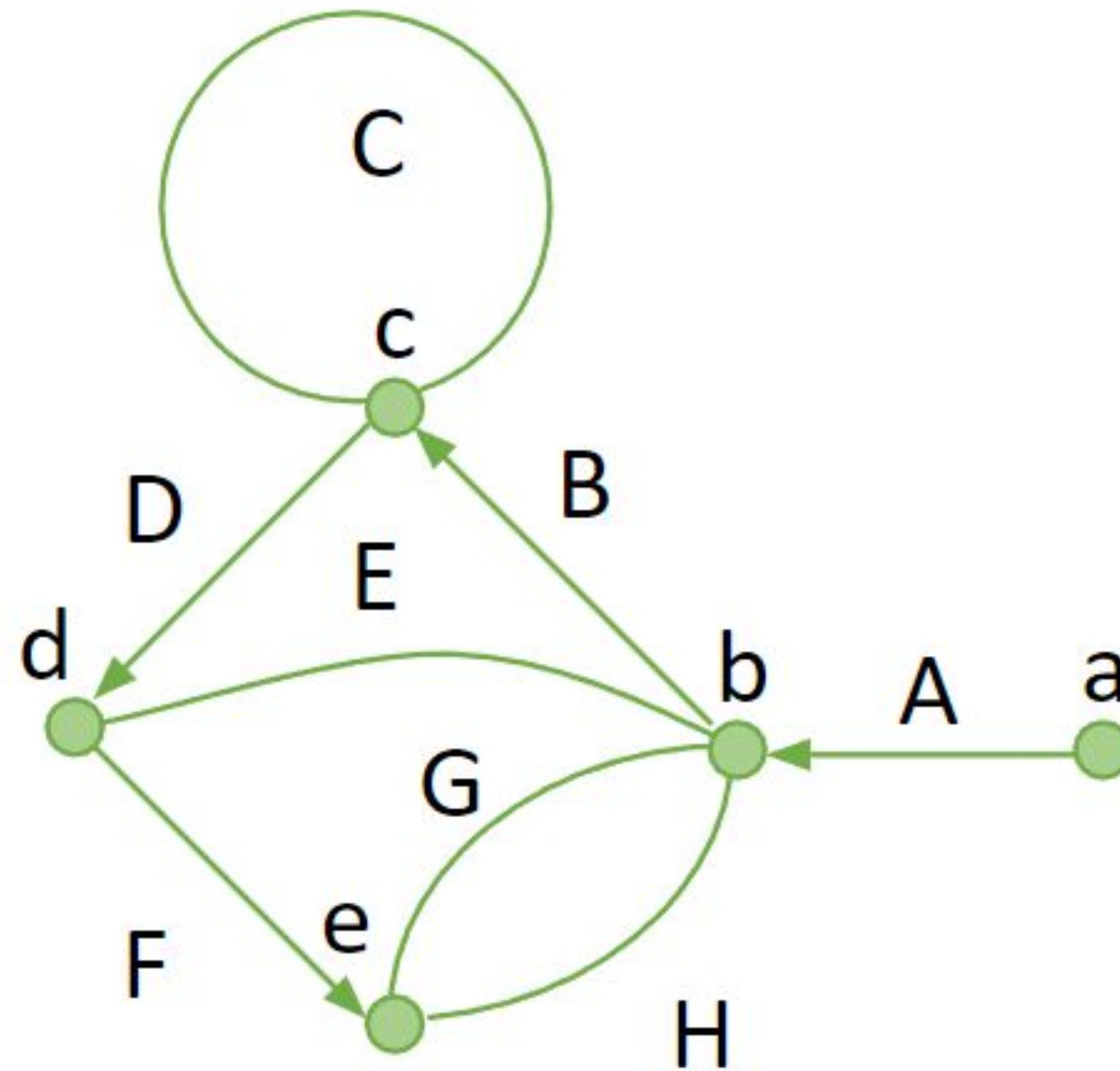


ou



$\Psi(A) = \{a, b\}$
$\Psi(B) = \{b, c\}$
$\Psi(C) = \{c, c\}$
$\Psi(D) = \{c, d\}$
$\Psi(E) = \{b, d\}$
$\Psi(F) = \{d, e\}$
$\Psi(G) = \{b, e\}$
$\Psi(H) = \{b, e\}$

Dígrafo



$$\Psi(A) = (a, b)$$

$$\Psi(B) = (b, c)$$

$$\Psi(C) = (c, c)$$

$$\Psi(D) = (c, d)$$

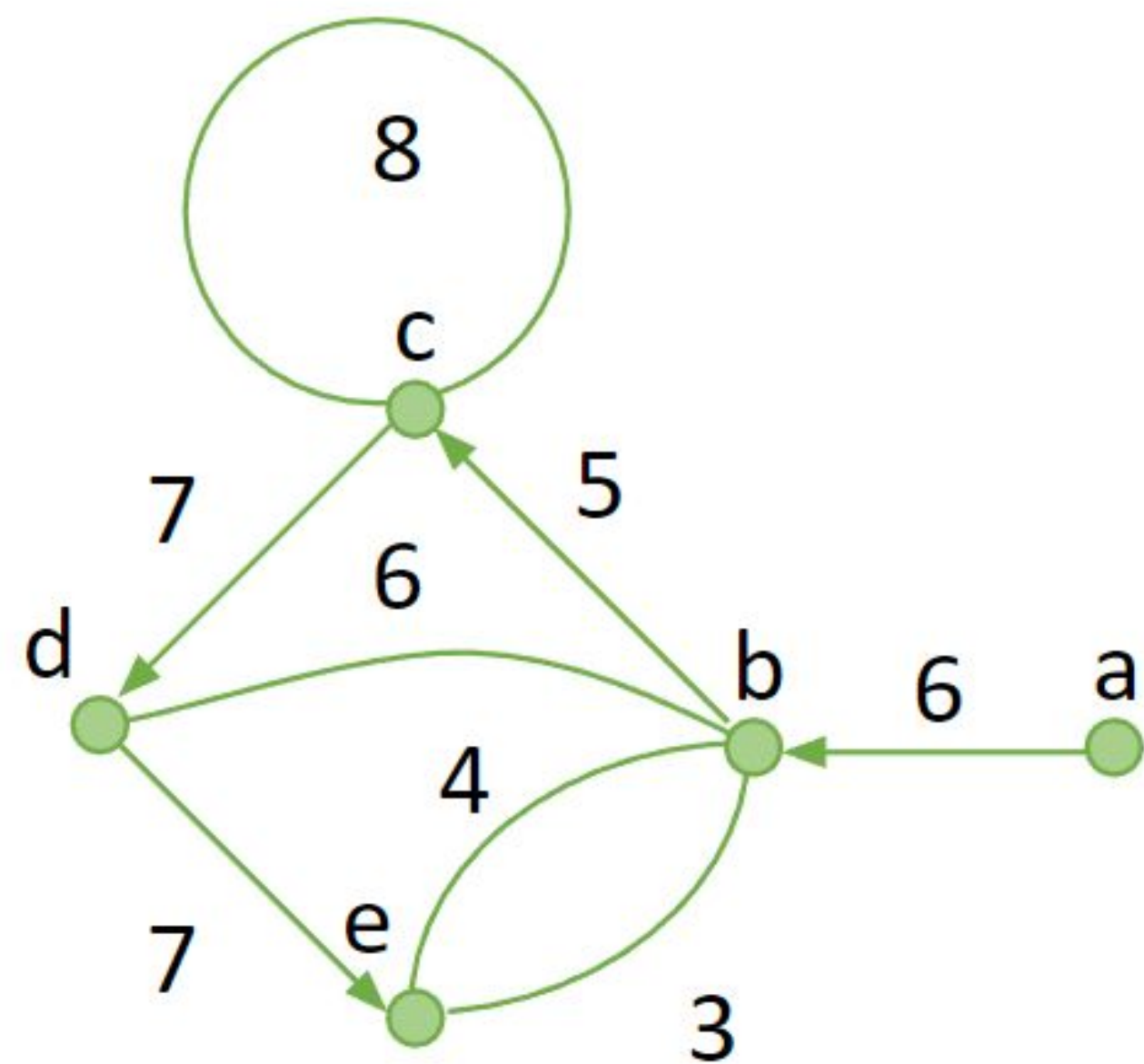
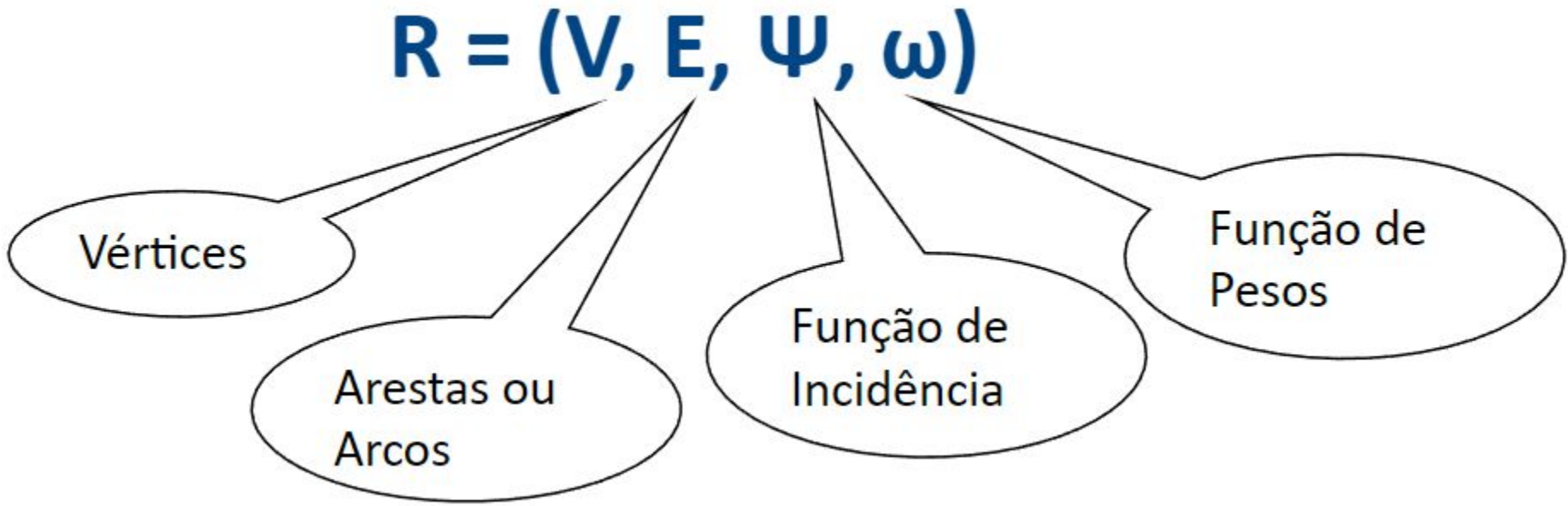
$$\Psi(E) = (b, d)$$

$$\Psi(F) = (d, e)$$

$$\Psi(G) = (b, e)$$

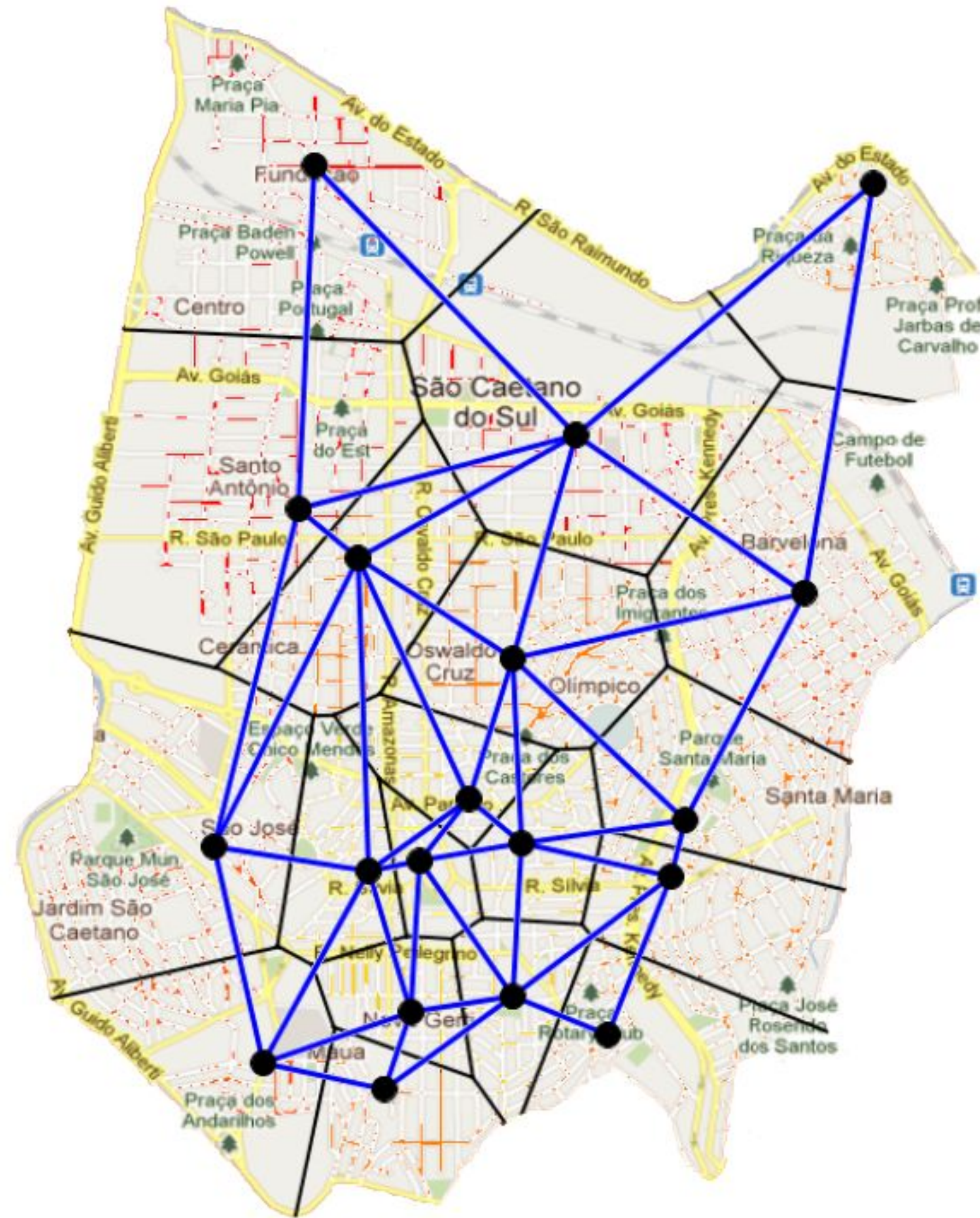
$$\Psi(H) = (b, e)$$

Grafo ponderado



$\Psi(A) = (a, b)$	$\omega(A) = 6$
$\Psi(B) = (b, c)$	$\omega(B) = 5$
$\Psi(C) = (c, c)$	$\omega(C) = 8$
$\Psi(D) = (c, d)$	$\omega(D) = 7$
$\Psi(E) = (b, d)$	$\omega(E) = 6$
$\Psi(F) = (d, e)$	$\omega(F) = 7$
$\Psi(G) = (b, e)$	$\omega(G) = 4$
$\Psi(H) = (b, e)$	$\omega(H) = 3$

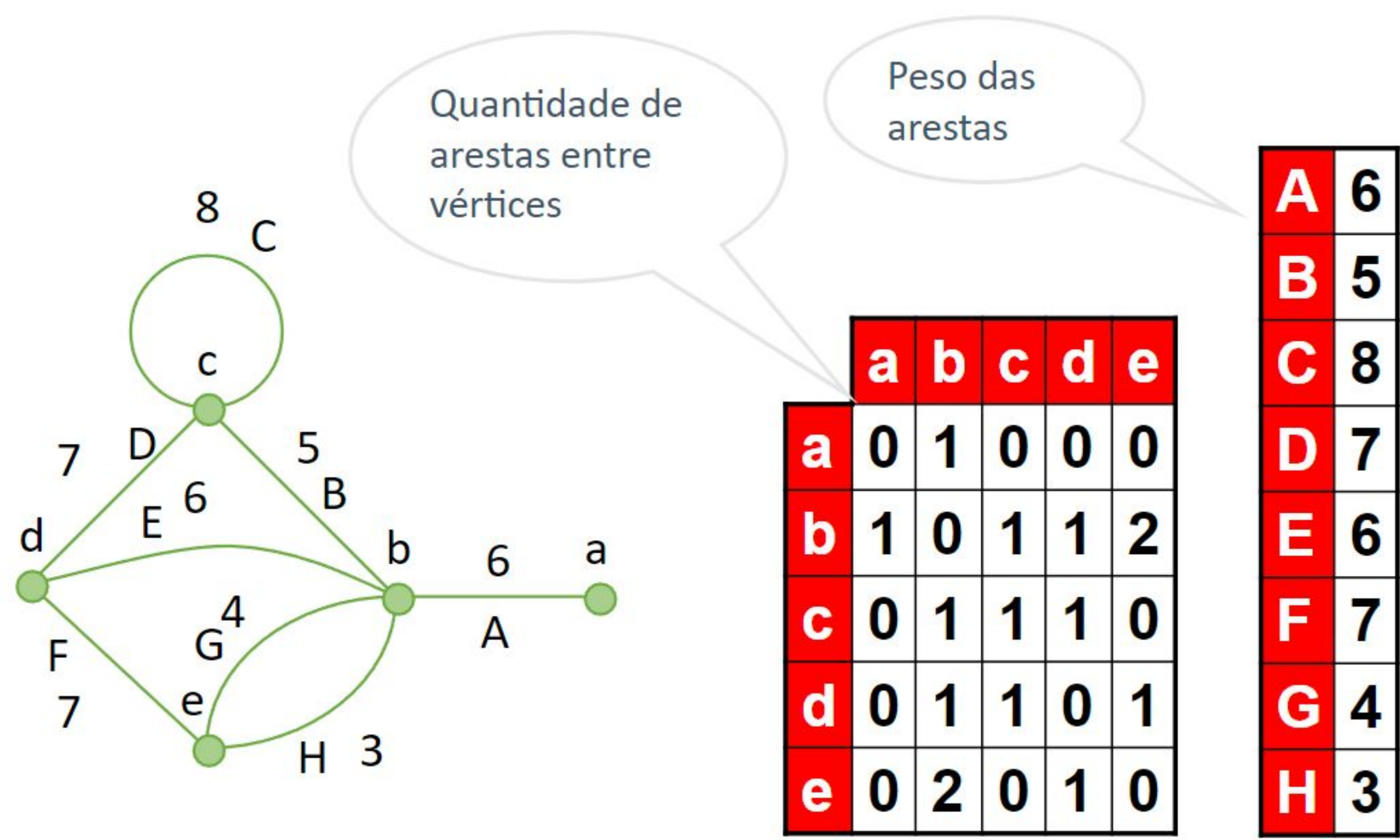
Exemplo de aplicação



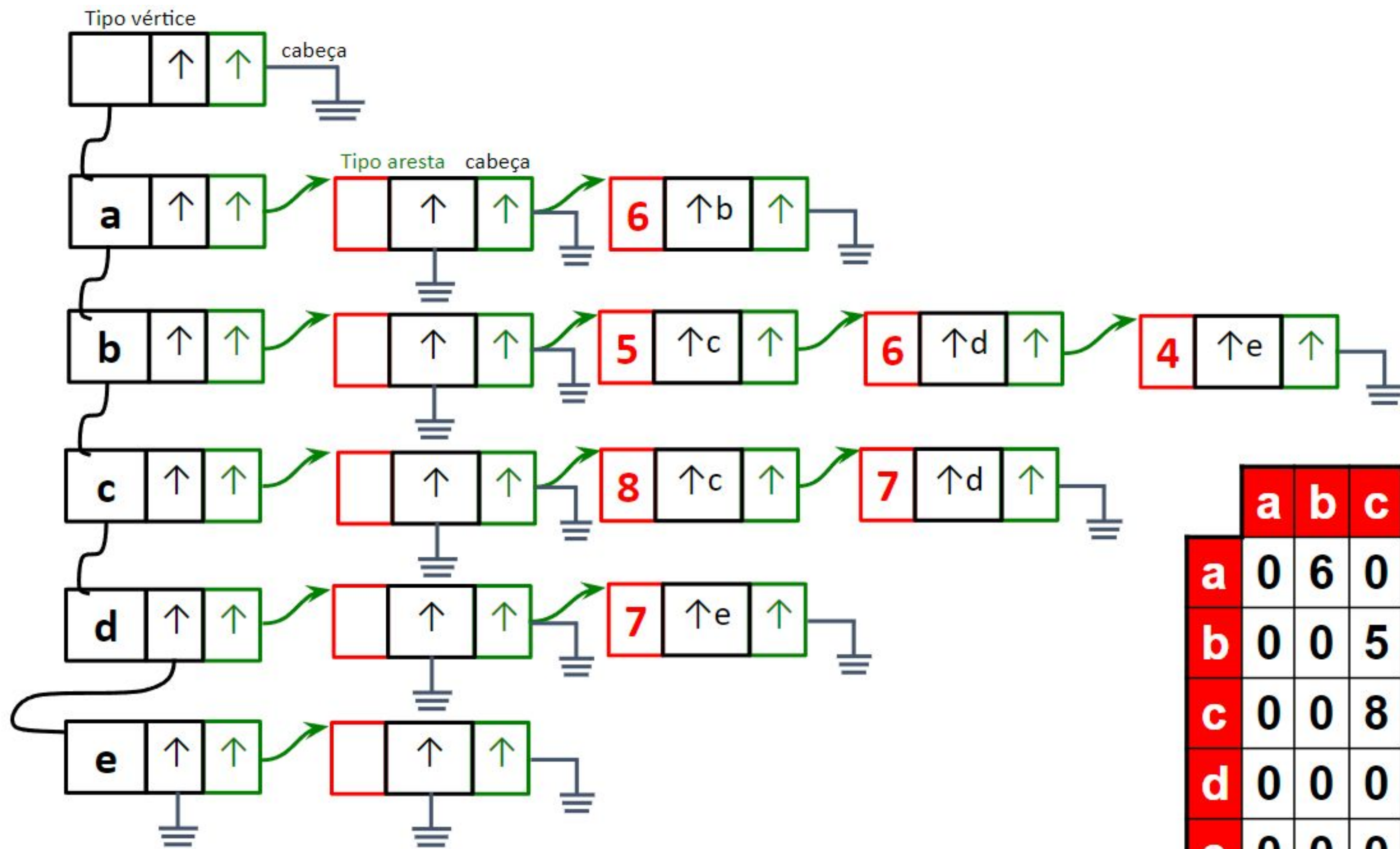
Representação de grafos



Matriz de adjacências



Lista de arestas



	a	b	c	d	e
a	0	6	0	0	0
b	0	0	5	6	4
c	0	0	8	7	0
d	0	0	0	0	7
e	0	0	0	0	0

Buscas em grafos

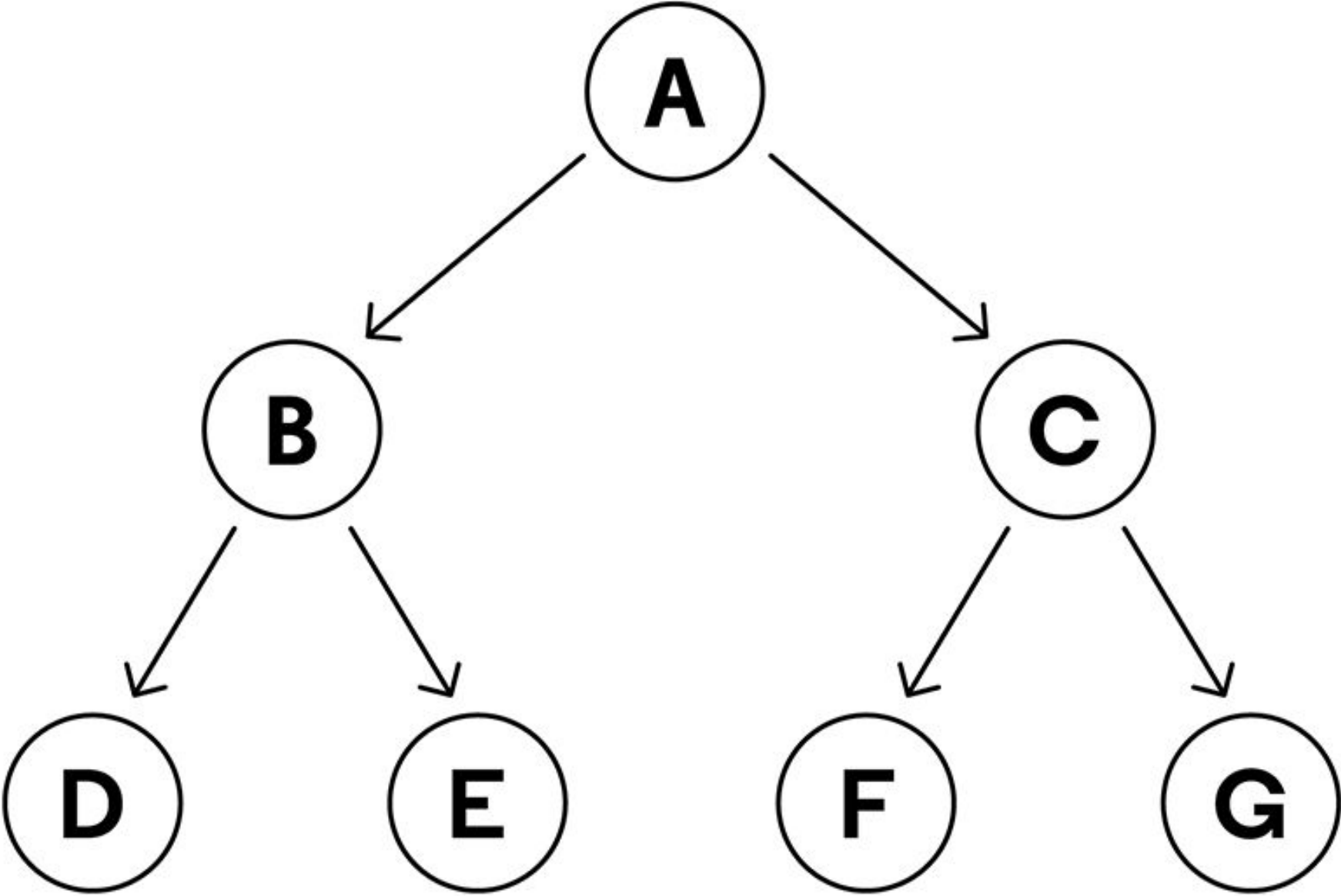


Busca em largura



Frontier Queue
FIFO (First in First Out)

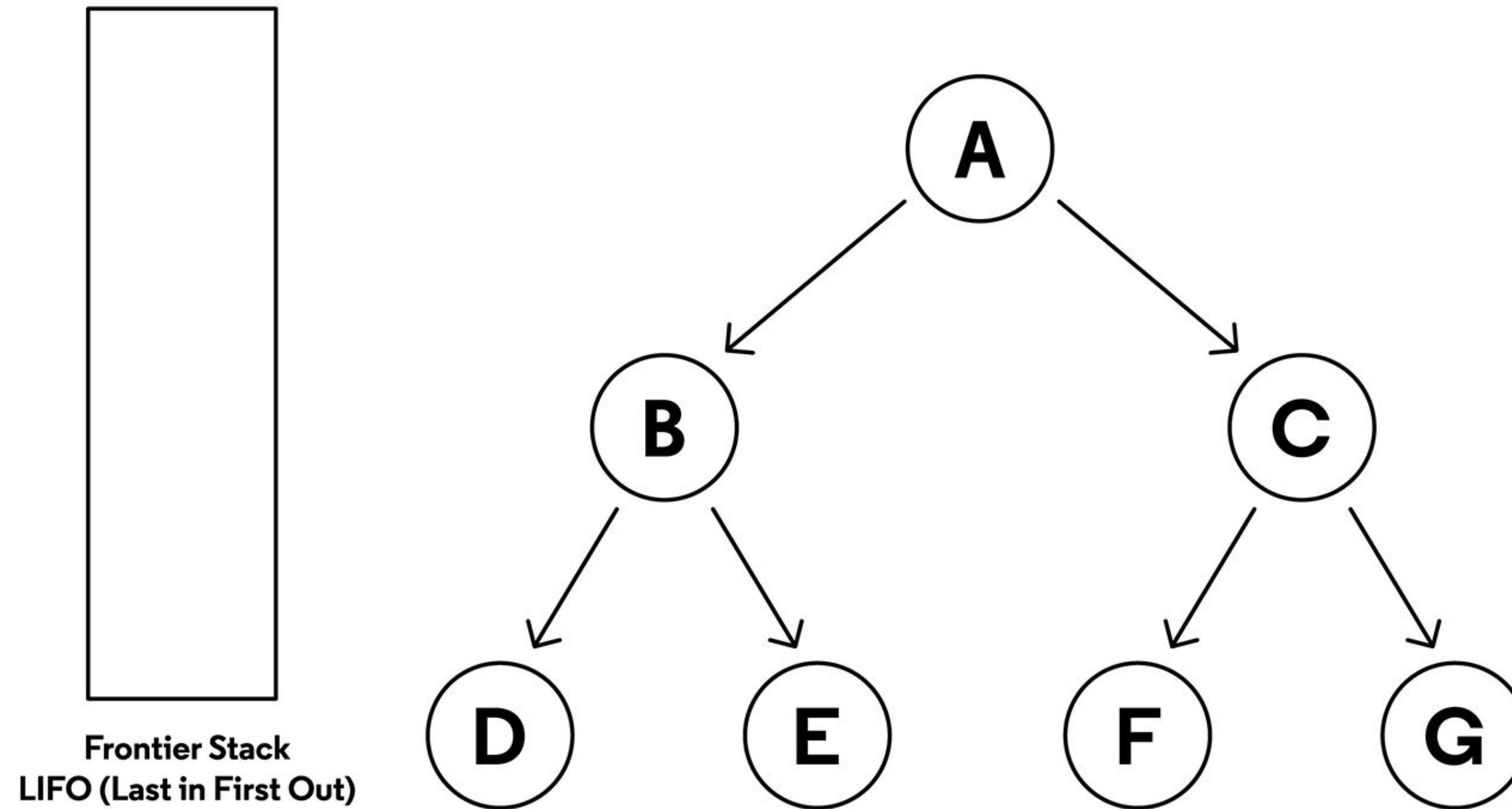
Tree with an Empty Queue



Retirado de: <https://www.codecademy.com/article/tree-traversal>

Busca em profundidade

Tree with an Empty Stack



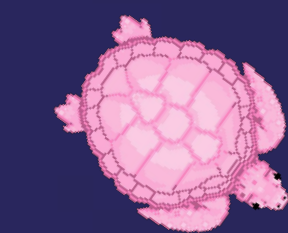
Retirado de: <https://www.codecademy.com/article/tree-traversal>

Outros algoritmos...

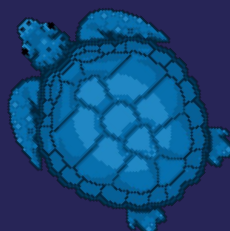
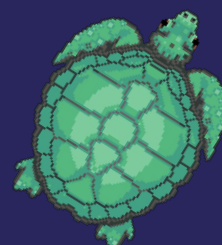
☐ Dijkstra

☐ Primm

☐ A*



Mão na massa



Um ajuste ao subscriber!

```
import rclpy
from rclpy.node import Node
from turtlesim.msg import Pose

class TurtleController(Node):
    def __init__(self, mission_control, control_period=0.01):
        super().__init__("turtlecontroller")
        self.pose_subscription = self.create_subscription(
            msg_type=Pose,
            topic='/odom',
            callback=self.pose_callback,
            qos_profile=10
        )

    def pose_callback(self, msg):
        self.get_logger().info(f"x={msg.x}, y={msg.y}, theta={msg.theta}")
```

Um ajuste ao subscriber!

```
import rclpy
from rclpy.node import Node
from nav_msgs.msg import Odometry
from tf_transformations import euler_from_quaternion

class TurtleController(Node):
    def __init__(self, mission_control, control_period=0.01):
        super().__init__("turtlecontroller")
        self.pose_subscription = self.create_subscription(
            msg_type=Odometry,
            topic="/odom",
            callback=self.pose_callback,
            qos_profile=10
        )

    def pose_callback(self, msg):
        x = msg.pose.pose.position.x
        y = msg.pose.pose.position.y
        z = msg.pose.pose.position.z
        ang = msg.pose.pose.orientation
        _, _, theta = euler_from_quaternion([ang.x, ang.y, ang.z, ang.w])
        self.get_logger().info(f"x={x}, y={y}, theta={theta}")
```

```
sudo apt install ros-humble-tf-transformations
```

```
pip install transforms3d
```

Sugestão

- ❑ Subscriber de pose
- ❑ Publisher de velocidades
- ❑ Controlador simples
- ❑ Lembrem-se dos vídeos! Disponíveis em <https://github.com/rmnicola/m6-ec-encontros>

Perguntas?



Retirado de
(<https://cdn-icons-png.flaticon.com/512/1268/1268705.png>), em 31/01/2023



Desenvolvimento

Retirado de
(<https://media.tenor.com/npYHSol-FgAAAAC/vegeta-training.gif>), em 31/01/2023

